ISSN:2184-044X ISBN:978-989-99864-3-5 © 2017

ENGAGING STUDENTS FOR THE LEARNING AND ASSESSMENT OF THE ADVANCED COMPUTER GRAPHICS MODULE USING THE LATEST TECHNOLOGIES

Yonghuai Liu[1], Longzhi Yang[2], Jiwan Han[3], Bin Lu[4], Peter Yuen[5], Yitian Zhao[6], & Ran Song[7]

[1]*Department of Computer Science, Aberystwyth University (UK)*
[2]*Department of Computer and Information Sciences, Northumbria University Newcastle (UK)*
[3]*Institue of Biological, Environmental and Rural Sciences, Aberystwyth University (UK)*
[4]*School of Computer Science and Technology, North China Electric Power University (China)*
[5]*Electro-Optics,Image & Signal Processing, Centre for Electronics Warfare, Cranfield University (UK)*
[6]*School of Optics and Electronics, Beijing Institute of Technology (China)*
[7]*School of Computing, Engineering and Mathematics, University of Brighton (UK)*

Abstract

The advanced computer graphics has been one of the most basic and landmark modules in the field of computer science. It usually covers such topics as core mathematics, lighting and shading, texture mapping, colour and depth, and advanced modeling. All such topics involve mathematics for object modeling and transformation, and programming for object visualization and interaction. While some students are not as good in either mathematics or programming, it is usually a challenge to teach computer graphics to these students effectively. This is because it is difficult for students to link mathematics and programming with what they used to see in video games and the TV advertisements for example and thus they can easily be put off. In this paper, we investigate how the latest technologies can help alleviate the teaching and learning tasks. Instead of selecting the low level programming languages for demonstration and assignment such as Java, Java 3D, C++, or OpenGL, we selected Three.js, which is one of the latest and freely accessible 3D graphics libraries. It has a unique advantage that it provides a seamless interface between the main stream web browsers and 2D/3D graphics. The developed code can be run on a web browser such as Firefox, Chrome, or Safari for testing, debugging and visualization without code changing. The unique design patterns and objectives of Three.js can be very attractive to third party software houses to develop auxiliary functions, methods and tutorials and to make them freely available for the public. Such a unique property of Three.js and its widely available supporting resources are especially helpful to engage students, inspire their learning and facilitate teaching.
To evaluate the effectiveness for using Three.js in teaching computer graphics we have set up an assignment for scene modeling in the last 4 years with focuses on the quality of the simulated scene (50%) and the quality of the assignment report (50%). We have evaluated different assessment forms of the module that we taught in the last four years: in 2013-2014 the module consisted of 20% assignment and 80% exam based on Java 3D; in 2014-2015 the same proportion of assignment/exam but based on WebGL, in 2015-2016 the module was 50-50% of assignment and exam but based on Three.js; and in this year the module is 100% assignment based on Three.js. The effectiveness of the module delivery has been evaluated both qualitatively and quantitatively from five aspects: a) average marks of students, b) moderator report, c) module evaluation questionnaire, d) external examiner's comments and e) examination board recommendations. The results have shown that Three.js is indeed more successful in engaging students for learning and the 100% assignment assessment enables students to focus more on the design and development. This four year result is really encouraging to us as an educational institute to embrace the latest technologies for the delivery of such challenging modules as computer graphics and machine learning.

Keywords: *Computer graphics module, Assessment, Assignment, Latest technologies, Student engagement.*

1. Introduction

Higher education has been considered as one of the most influential factors to the future of youngsters. Regardless of the particular motives for individuals to take part in the university education, one of the most common goals is to learn something from the university that may be useful for their later life. However, to meet this basic requirement/goal is actually very challenging! This is because various

factors, such as the students' personal backgrounds and commitments, together with the changing and abstract nature of the intended study, delivery, support and assessment by the course would give variable results. From the university perspective it is important to ensure all means to help design and deliver the module and facilitate the learning process as much as possible.

Technology has been a widely used tool to enhance the teaching and learning of science in various disciplines. For example, videos have been used in (Alkhalaileh, Hasan, Al-Rawajfah, 2017) for instructing the skills of the Cardio-Pulmonary resuscitation to medical students. It is found that the method is as effective as traditional class form lecture format of teaching. A mobile system has been developed (Mohamed, Chebbi, Behera, 2016) to help students to learn at any time and locations at their own paces. The study reported by (Chowdhry, Sieler, Alwis, 2014) has shown that it is necessary to continuously improve the method for technology-enhanced teaching and also the learning skills of staff in order to enhance the student learning efficiency.

In this paper, we investigate the main issues in the delivery and assessment of the advanced computer graphics module enhanced by the latest open source library Three.js. While such module finds numerous applications in the real world such as computer games, data analysis and visualization, and specialized effects in the films and TV advertisements, the topics are usually abstract for students to link the course to the real world. With the technological development in programming and the availability of high computational power, various new programming languages such as Java3D and WebGL and open source libraries such as Three.js have been developed. Interestingly enough, various routines in the creation of a computer graphics system can be encapsulated without being changed from one application to another, so that the end user can just focus on the core tasks such as object modeling.

Three.js is an open source 3D graphics library, whose code can be run seamlessly on the mainstream web browsers such as Chrome, Mozilla Firefox, and Safari. It provides an intuitive application programming interface (API) between the 3D graphics, web browsers, and hardware. In this case, we use it for topics explanations, demonstrations, practicals, and assignment. With each main topic covered in the class such as core mathematics (e.g. trigonometry, matrix and vector algebra), lighting and shading, texture mapping, buffers, colour and depth, advanced modeling and ray tracing, (i) various demonstration programs have been developed in Three.js to show the effects generated, (ii) a practical of up to two hours is designed after each topic so that students can see the basic implementation of a relative system and then modify, implement and complete the required tasks, (iii) an assignment is setup based on the topics covered and the practical experimentations. One session has also been designed for answering questions two weeks before the deadline for the assignment submission.

To evaluate the effectiveness of the Three.js for learning and teaching, a comparative study is carried out in the last four years based on Java3D and WebGL assessed with different weights between assignments and examinations. The evaluation is carried out from different aspects: the average marks of the class, feedback from the middle term module evaluation questionnaire (MEQ), and comments from the external examiner, the module moderator, and the examination board.

2. Content design and delivery

In this section the design and delivery of the contents in the module is outlined. While the course normally covers topics such as lighting, shading and texture mapping; it is crucial to create a virtual world for the visualization, demonstration and interaction of the objects of interest.

2.1. A general framework of a computer graphics system

A general framework for the creation of a computer graphics system can be built below using Three.js (Yadav, 2015; Liu, Liu, Zhao, Song, 2016) as:

```
<html>
	<head>
		<title>My first Three.js app</title>
		<style>
			body { margin: 0; }
			canvas { width: 100%; height: 100% }
		</style>
<script src="js/three.min.js"></script>
	</head>
	<body>
		<script>
			// Our Javascript will go here.
		</script>
	</body>
</html>
```

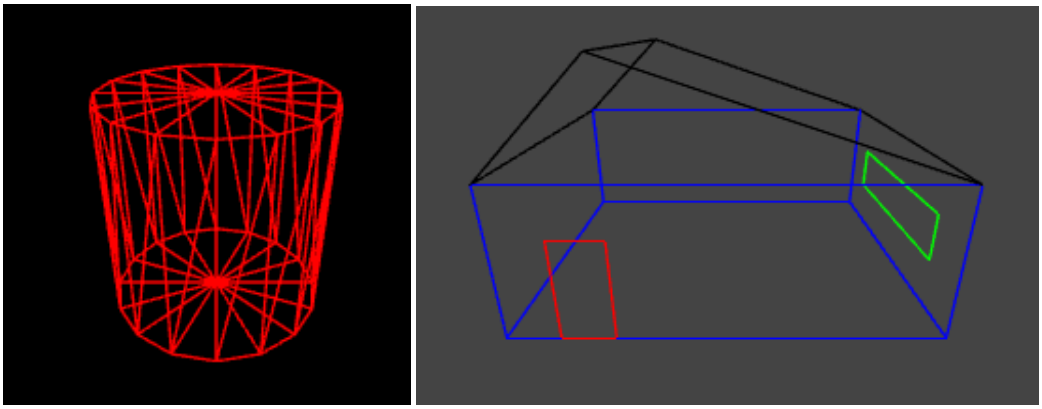
This framework can be stored as a file called my3d.html in the current directory, c:\graphics, which can then be run on the commonly used web browsers such as Chrome and Mozilla Firefox: file:///c:/graphics/my3d.html. It is a modified version of the standard html document file (HTML, 2016), starting with a tag of <html> and ending with a tag of </html>. It includes two parts, head and body. The head element is a container of metadata and typically defines the document title, styles, links, scripts, and other metadata. It has a start tag of <head> and an end tag of </head>. The body element defines the document body, and has a start tag of <body> and an end tag of </body>. The line "<script src="js/three.min.js"></script>" tells the browser that the minimized version of Three.js has been stored in the subdirectory js and will be called in and used for interpreting the subsequent code. Then the 3D contents will be created and inserted into the body of the document for visualization and interaction.

2.2. Example content generation

To actually display an object in the above framework, we need to define three basic objects: camera, renderer, and scene. The scene describes the objects and their living world. The camera defines how the scene will be visualized. The renderer projects the scene onto the image plane of the camera. All such objects include varying numbers of variables for their specification and manipulation.

For example, we define two variables for the size of the canvas onto which the scene will be drawn as: var width=window.innerWidth/2; var height=window.innerHeight/2 as half of the width and height of the current viewing window. The renderer is defined as a variable: var renderer = new THREE.WebGLRenderer({antialias: true}); as a new instance of the class THREE.WebGLRenderer with the variable antialias taking the value of "true" so that the boundaries of the rendered objects will be smoothed. The size of the renderer is set as: renderer.setSize(width, height); using the variables width and height just defined above. The rendered objects need to be inserted into the body of the html file for visualization as document.body.appendChild(renderer.domElement); The scene is defined as an instance of the class THREE.Scene: var scene = new THREE.Scene(); The camera is defined as a variable: var camera = new THREE.PerspectiveCamera(75, width / height, 0.1, 1000); as an instance of the class of THREE.PerspectiveCamera taking four parameters: 75 is the angle for the vertical field of view, width/height defines the aspect ratio of the camera frustum, 0.1 and 1000 define the closest and farthest planes between which the objects will be rendered and displayed. All other objects, either too close or too far away from the camera, will not be rendered and displayed. Then the camera is moved away from the origin of (0, 0, 0) along the z axis by 20 units as: camera.position.set(0, 0, 20); Finally, the defined camera is inserted into the scene as: scene.add(camera); So far the virtual world has been created and a computer graphics system has been set up.

Figure 1. The visualization of the cylinder(left) and house(right) designed.



Suppose that a cylinder will be created and displayed in the virtual environment defined above. An object has to be defined from two aspects as an instance of the class THREE.Mesh in Three.js: geometry and material. The geometry defines the geometrical description of the object of interest, while the material defines how the object will look like. A primitive THREE.CylinderGeometry has been defined in Three.js. In this case, we can directly call it our application as: var geometry = new THREE.CylinderGeometry(5, 5, 10, 16); taking four parameters: specifying the radius of the top and bottom, height and the number of segments in the top and bottom of the cylinder. The material is defined as a variable: var material = new THREE.MeshBasicMaterial({ color: 0xff0000, wireframe: true }); as a new instance of the class THREE.MeshBasicMaterial with a color of red represented as 0xff0000 in the hexadecimal format and rendered as wireframe. Then the cylinder is defined as var cylinder = new THREE.Mesh(geometry, material); In order to see the top of the cylinder, we rotate it along the x axis by 30 degrees as: cylinder.rotation.x = Math.PI * 30 / 180; where the rotation angle has to be represented in radians. The cylinder is added into the scene for display: scene.add(cylinder); Finally, the method

renderer.render(scene, camera) is called to project the cylinder onto the canvas by the renderer for visualization as illustrated in Figure 1.

From the above discussion, it can be seen that it is relatively easy to define with intuitive commands the environment, camera, and rendering process, and add objects into the scene for visualization. All such operations have been encapsulated so that the particular applications do not require much time on such routine works but focus on the core tasks such as cylinder modeling instead.

2.3. Advanced content generation

In this section, we demonstrate how Three.js can be applied to implement a complicated object, a house in wireframe, in this case, in which various classes, methods, transformations, configurations, and representation will be involved in an easy to understand and follow manner. First, we have to define a variable: var house= new THREE.Object3D(); as an instance of the class of THREE.Object3D to hold its components: front wall, right wall, left wall, rear wall, right roof, left roof, front window and right window and add it into the scene as scene.add(house); Each of these components is represented as a rectangle. The corners of these rectangles have to be designed carefully, considering their desired sizes and locations as: var vertices = [new THREE.Vector3(-4, -3, 0), new THREE.Vector3(4, -3, 0), new THREE.Vector3(4, -3, 3), new THREE.Vector3(-2, -3, 5), new THREE.Vector3(-4, -3, 3), new THREE.Vector3(-4, 3, 3), new THREE.Vector3(-2, 3, 5), new THREE.Vector3(4, 3, 3), new THREE.Vector3(4, 3, 0), new THREE.Vector3(-4, 3, 0), new THREE.Vector3(-3, -3, 0), new THREE.Vector3(-2, -3, 0), new THREE.Vector3(-2, -3, 2), new THREE.Vector3(-3, -3, 2), new THREE.Vector3(4, -2, 1), new THREE.Vector3(4, 2, 1), new THREE.Vector3(4, 2, 2), new THREE.Vector3(4, -2, 2)]; Then these vertices have to be associated with a particular component.

Firstly, we define geometry and material variables for the representation of all these components as new instances of the classes THREE.Geometry and THREE.LineBaiscMaterial respectively. For the front wall, they are defined as: var lineGeom1=new THREE.Geometry(); var lineMat1=new THREE.LineBasicMaterial({color: 0x0000ff }), leading the front wall to be defined as: var face1=new THREE.Line(lineGeom1, lineMat1) as a wireframe in blue; then we add it into the house as: house.add(face1); To instantiate lineGeom1, we have to provide the vertex information as: lineGeom1.vertices.push(vertices[0]); lineGeom1.vertices.push(vertices[1]); lineGeom1.vertices.push(vertices[2]); lineGeom1.vertices.push(vertices[4]); lineGeom1.vertices.push(vertices[0]); which essentially define a closed rectangle with the first, second, third, and fifth vertex already defined above; All the walls, roofs and windows can be defined similarly but with different sets of vertices (1, 8, 7, 2), (0, 4, 5, 9), (5, 7, 8, 9), (2, 7, 6, 3), (3, 6, 5, 4), (10, 11, 12, 13), and (14, 15, 16, 17). Note that some vertices are shared by different walls and roofs. Thus, they must be specified correctly. Suppose that the roof is represented in black, the front window is in red, and the right window is in green, then their materials are defined as: var lineMat5=new THREE.LineBasicMaterial({color: 0x000000}); var lineMat7=new THREE.LineBasicMaterial({color: 0xff0000}); and var lineMat8=new THREE.LineBasicMaterial({color: 0x00ff00}); respectively.

In order to see the top of the house, it is first moved away from the original (0, 0, 0) by -2 units along the y axis as: house.position.y = -2; then it is rotated by -70 degrees along the x axis as: house.rotation.x = -Math.PI*70/180; The built house is illustrated in Figure 1. From the above discussion, it can be seen that: (i) all the details of the object of interest must be designed clearly and described explicitly, (ii) the designed models can be translated into the language of Three.js relatively straightforward for visualization.

3. Assessment and evaluation

In this year the module was assessed purely based on the assignment: scene modeling and navigation. The students submitted both the source code and a report describing the contents of the scene and explaining how the objectives were achieved. Each element is worth 50% assessment of the module. The quality of the scene was marked from three aspects: complexity (20%), functionality (20%) and creativity (10%), while the report was marked from three aspects as well: description (15%), discussion (30%), and presentation (5%) of the report.

The performances of the classes in different years are presented in Table 1: clearly, the number of students registered onto the module increases over the last four years. Even so, the average performance of the class has not been dragged down. On the contrary, it has been increased steadily due to various factors, one of which is the adoption of the latest technologies of Three.js for the effective design and delivery of the model contents. Also, it is noted that the outstanding performance of the class this year in comparison to previous three years, is because the module was assessed purely based on assignment. Thus, we can conclude that the pure assignment based assessment enabled students to focus on the design, implementation, and test of the system without being distracted by the exam.

Table 1. The performance of the class in different academic years.

Academic year	#(students)	Assign/exam(%)	Average (%)	Std. dev. (%)	Failure rate (%)
2016-2017	57	100/0	64.56	12.48	1.78
2015-2016	39	50/50	59.10	17.77	10.26
2014-2015	34	20/80	59.68	15.84	8.80
2013-2014	14	20/80	58.00	17.18	7.10

Feedbacks with positive comments like: "The module was well run and students seemed to engage well with the assignment. It might be worth briefly mentioning coding style, and especially use of global variables in the JavaScript that is available for use by the students." from the moderator, "Really good well taught module!", "Feedback has been well acted on", and "Some of the lectures have a heavy mathematical focus which although useful, can end up being a bit overwhelming sometimes." from the middle term MEQ this year. Some comments from the middle term MEQ in 2015-2016 are: "The module is very good structurally, could use a bit harder material such as OpenGl." The comments from the external examiner are: "a large amount of computation involved in the answers. The questions are in the right level." in 2015-2016 and "General: a large amount of contents involved in the 2 hour examination, but only three questions answered from five. The question level is fine." in 2014-2015. All the results have been accepted as a set by the departmental examination board.

4. Conclusions

This paper investigates two main issues: the delivery and assessment of an advanced computer graphics module. To facilitate the delivery of the contents in the module, we adopted the latest open source library Three.js as the main language for topics explanation, demonstration, practical experimentation and assignment. The package has an advantage that normal routine functions are encapsulated, various primitives are provided, the code can be easily tested on the main stream web browsers and students can thus just focus on the core tasks such as object modeling. The module was assessed purely on an assignment with a submission of two elements: source code and a report so that students can just focus on the design, implementation, test and report writing without being distracted by the exam. Various means have been employed to evaluate the effectiveness of such "technology enhanced learning and assignment only" course. A comparative study of the learning and teaching in the last four years has shown that this approach does enhance the efficiency of learning and teaching of the module and leads to a positive change in the attitude of students to learn (Kirkwood and Price, 2014). This is demonstrated by the fact that the students picked up the topics more quickly, were more capable to develop a system for testing, and dedicated themselves to improve the creativity of the system more eagerly. We are thus encouraged to embrace the latest technologies for the teaching and delivery of such challenging modules as computer graphics and machine learning.

References

Kirkwood, A.; Price, L. (2014). Technology-enhanced learning and teaching in higher education: what is 'enhanced' and how do we know? A critical literature review. *Learning, Media and Technology, 39(1)*, pp. 6-36.

Alkhalaileh, M., Hasan, A.A., Al-Rawajfah, O. (2017). Evaluate the effectiveness of clinical simulation and instructional video training on the nursing students' knowledge about Cardio-Pulmonary resuscitation: comparative study. *American Journal of Educational Research, 5(1)*, pp. 63-68.

Mohamed S., Chebbi, M., Behera. (2016). Pervasive mobile learning system in the new millennium. *American Journal of Educational Research, 4(18)*, pp. 1257-1264.

Chowdhry, S., Sieler, K., Alwis, L. (2014) A study of the impact of technology-enhanced learning on student academic performance. *Journal of Perspectives in Applied Academic Practice, 2(3*), pp. 3-15.

three.js/docs. (n.d.). Three.js documentation, http://threejs.org/docs/

Yadav, A. (2015). Creating 3D cube: a practical guide to three.js with live demo, http://www.awwwards.com/creating-3d-cube-a-practical-guide-to-three-js-with-live-demo.html

Liu, Y., Liu, H., Zhao, Y., Song R. (2016) Teaching of advanced computer graphics with three.js. *Proceedings of International Conference on Education and New Developments*, pp. 13-17.

HTML. (2016). The language for building web pages, http://www.w3schools.com/html/html_head.asp